# Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

**Bài 1.** Có một phòng họp trong một công ty. Có N cuộc họp được đăng ký ở dạng (start [i], end [i]) trong đó start[i] là thời gian bắt đầu cuộc họp thứ i và end[i] là thời gian kết thúc cuộc họp thứ i.

Nhiệm vụ:

- Tạo vector để lưu trữ các cuộc họp đã cho dưới dạng **struct Interval**

- Viết hàm **sắp xếp** các cuộc họp trong vector trên theo **thứ tự tăng dần thời gian bắt đầu**.

- Các cuộc họp được đăng ký sử dụng trong các khoảng thời gian nào?

Ví dụ 1:

|  |
| --- |
| **Input:**  start = {1,3,0,8,5}  end = {5,4,6,9,9}  **Output:**  0 9 |

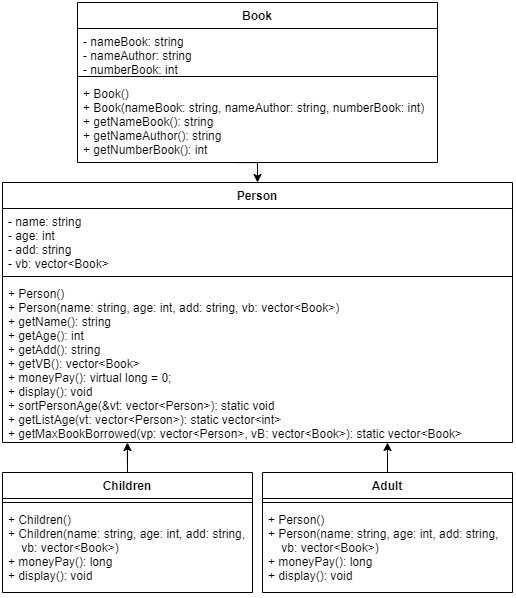
Ví dụ 2:

|  |
| --- |
| **Input:**  start = {1,3,0,8,8}  end = {5,4,3,10,9}  **Output:**  0 5  8 10 |

## **Bài 2.** **Bài 2.1.** Bạn hãy tạo 4 lớp Book, Person, Children, Adult với các thông tin như bên dưới.

**Bài 2.2.** Viết hàm main để kiểm thử tính chính xác của chương trình:

* Khai báo và khởi tạo một mảng hoặc vector 5 đối tượng lớp Book lưu các thông tin về 5 quyển sách. Thông tin về các đối tượng Book có thể được khởi tạo bằng giá trị ngẫu nhiên bất kỳ và hợp lệ sử dụng hàm tạo có tham số.
* Khai báo mảng chứa 3 đối tượng lớp Children và mảng chứa 2 biến đối tượng lớp Adult. Thông tin về các đối tượng Book được tạo ngẫu nhiên thông qua các hàm tạo có tham số tương ứng để khởi tạo dữ liệu cho thành phần của mảng.
* Gọi phương thức lớp display để hiển thị thông tin các đối tượng
* Tạo một vector 5 con trỏ kiểu Person trong đó chứa địa chỉ của 3 đối tượng lớp Children và 2 đối tượng lớp Adult được tạo bên trên.
* Gọi phương thức tĩnh của lớp Person trên vector bên trên và hiển thị kết quả.
  + sortPersonAge(&vt: vector<Person\*>)
  + getListAge(vt: vector<Person\*>): static vector<int>
  + getMaxBookBorrowed(vp: vector<Person\*>, vB: vector<Book>): static vector<Book>



Giải thích:

**Lớp Book:**

* nameBook, nameAuthor, numberBook lần lượt là các thuộc tính chỉ tên, tác giả, số lượng sách có phạm vi truy cập private.
* Book() là phương thức khởi tạo không tham số.
* Book(nameBook: string, nameAuthor: string, numberBook: int) là phương thức khởi tạo 3 tham số.
* display(): void là phương thức hiển thị thông tin Book ra màn hình. Ví dụ nameBook = "O Alquimista", nameAuthor = "Paulo Coelho", numberBook = 1000 thì sau khi gọi phưng thức display() màn hình sẽ hiển thị:
  + Ten sach: O Alquimista
  + Ten tac gia: Paulo Coelho
  + Ma so sach: 1000
* Các phương thức còn lại là các getter và setter của các thuộc tính tương ứng.

**Lớp Person:**

* Person là lớp trừu tượng thể hiện cho một độc giả, ở đây các lớp Children và Adult kế thừa class Person để chỉ độc giả là trẻ em và người lớn.
* name, age, add, vb lần lượt là các thuộc tính chỉ tên, tuổi, địa chỉ và danh sách các cuốn sách mượn của độc giả có phạm vi truy cập private.
* Person() là phương thức khởi tạo không tham số.
* Person(name: string, age: int, add: string, vb: vector<Book>) là phương thức khởi tạo 4 tham số.
* moneyPay(): virtual long = 0 là phương thức thuần ảo tính toán số tiền phải trả của độc giả.
* display(): là *phương thức ảo* để hiển thị thông tin Person. Các thông tin hiện ra gồm thông tin cá nhân và danh sách sách mượn.
  + Name: Nam
  + Age: 30
  + Add: Ha noi
  + List Book Borrowed
    - Cuon sach 1:
    - Name Book: O Alquimista
    - Name Author: Paulo Coelho
    - Number Book: 1000
    - Cuon sach 2:
    - Name Book: Hello World
    - Name Author: C++
    - Number Book: 1002
* sortPersonAge(&vt: vector<Person\*>): static void là ***phương thức tĩnh*** sắp xếp các độc giả theo thứ tự tăng dần của tuổi.
* getListAge(vt: vector<Person\*>): static vector<int> là ***phương thức tĩnh*** trả về một vector chứa độ tuổi không lặp của các độc giả. Ví dụ có tuổi 14, 14, 15, 16, 15 thì trả về một vector chứa các phần tử 14, 15, 16.
* getMaxBookBorrowed(vp: vector<Person\*>, vB: vector<Book>): static vector<Book> là phương thức tĩnh trả về danh sách các cuốn sách được mượn nhiều nhất.

**Lớp Children:**

* Children là lớp thể hiện cho độc giả trẻ em kế thừa public của class Person.
* Children() là phương thức khởi tạo không tham số.
* Children(name: string, age: int, add: string, vb: vector<Book>) là phương thức khởi tạo 4 tham số.
* moneyPay(): long là phương thức trả về số tiền phải trả khi mượn sách theo công thức (số sách mượn)\*5000.
* display(): void là phương thức hiển thị thông tin của độc giả trẻ em. Phương thức này hiển thị dòng chữ “Doc gia Tre em” sau đó là các thông tin khác là thông tin mà phương thức display() của lớp Person hiển thị.

**Lớp Adult:**

* Adult là lớp thể hiện cho độc giả người lớn kế thừa public của class Person.
* Adult() là phương thức khởi tạo không tham số.
* Adult(name: string, age: int, add: string, vb: vector<Book>) là phương thức khởi tạo 4 tham số.
* moneyPay(): long là phương thức trả về số tiền phải trả khi mượn sách theo công thức (số sách mượn)\*10000.
* display(): void là phương thức hiển thị thông tin của độc giả người lớn. Phương thức này hiển thị dòng chữ “Doc gia Nguoi lon” sau đó là các thông tin khác là thông tin mà phương thức display() của lớp Person hiển thị.

# 

# Đáp án - Đề thi cuối kỳ

Môn : Lập trình C nâng cao

Thời gian: 90 phút

# 

#### Person.cpp

#pragma once

#include <vector>

#include <algorithm>

#include "Book.cpp"

class Person

{

private:

string name;

int age;

string add;

vector<Book> vb;

public:

Person() {}

Person(string name, int age, string add, vector<Book> vb) : name(name), age(age), add(add), vb(vb) {}

string getName()

{

return name;

}

int getAge()

{

return age;

}

string getAdd()

{

return add;

}

vector<Book> getVB()

{

return vb;

}

virtual long moneyPay() = 0;

virtual void display()

{

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;

cout << "Add: " << add << endl;

cout << "List Book Borrowed:" << endl;

for (int i = 0; i < vb.size(); i++)

{

vb[i].display();

}

cout << "Money Pay: " << moneyPay() << endl;

}

static void sortPersonAge(vector<Person \*> &vt)

{

for (int i = 0; i < vt.size() - 1; i++)

{

for (int j = i + 1; j < vt.size(); j++)

{

if (vt[i]->getAge() > vt[j]->getAge())

{

swap(vt[i], vt[j]);

}

}

}

}

static vector<int> getListAge(vector<Person \*> vt)

{

vector<int> str;

for (int i = 0; i < vt.size(); i++)

{

if (!(find(str.begin(), str.end(), vt[i]->getAge()) != str.end()))

{

str.push\_back(vt[i]->getAge());

}

}

return str;

}

static vector<Book> getMaxBookBorrowed(vector<Person \*> vp, vector<Book> vB)

{

vector<int> count(vB.size(), 0);

for (int i = 0; i < vB.size(); i++)

{

for (int j = 0; j < vp.size(); j++)

{

vector<Book> b(vp[j]->getVB());

for (int k = 0; k < b.size(); k++)

{

if (vB[i].getNameBook() == b[k].getNameBook())

{

count[i]++;

}

}

}

}

int max = \*max\_element(count.begin(), count.end());

vector<Book> res;

for (int i = 0; i < vB.size(); i++)

{

if (count[i] == max)

{

res.push\_back(vB[i]);

}

}

return res;

}

};

#### Children.cpp

#pragma once

#include "Person.cpp"

class Children : public Person

{

public:

Children() {}

Children(string name, int age, string add, vector<Book> vb) : Person(name, age, add, vb) {}

long moneyPay()

{

return this->getVB().size() \* 5000;

}

void display()

{

cout << "Children!" << endl;

Person::display();

}

};

#### Adult.cpp

#pragma once

#include "Person.cpp"

class Adult : public Person

{

public:

Adult() {}

Adult(string name, int age, string add, vector<Book> vb) : Person(name, age, add, vb) {}

long moneyPay()

{

return this->getVB().size() \* 10000;

}

void display()

{

cout << "Adult!" << endl;

Person::display();

}

};

### Bài 2.2.

int main()

{

vector<Book> books(5);

books[0] = Book("A", "A", 1);

books[1] = Book("B", "B", 2);

books[2] = Book("C", "C", 3);

books[3] = Book("D", "D", 4);

books[4] = Book("E", "E", 5);

Children children[3];

children[0] = Children("A", 10, "HN", {books[0], books[1]});

children[1] = Children("B", 20, "HN", {books[2], books[4], books[3]});

children[2] = Children("C", 30, "HN", {books[3], books[4]});

Adult adults[2];

adults[0] = Adult("A", 1, "HN", {books[0], books[1]});

adults[1] = Adult("B", 2, "HN", {books[0], books[1], books[2], books[3], books[4]});

vector<Person \*> persons(5);

persons[0] = &children[0];

persons[1] = &children[1];

persons[2] = &children[2];

persons[3] = &adults[0];

persons[4] = &adults[1];

cout << "Danh sach tuoi" << endl;

for (auto v : Person::getListAge(persons))

cout << v << " ";

cout << endl

<< "Sach muon nhieu nhat" << endl;

for (auto v : Person::getMaxBookBorrowed(persons, books))

v.display();

cout << "Sap xep theo tuoi" << endl;

Person::sortPersonAge(persons);

for (auto v : persons)

v->display();

}